Attorney Docket No.: 200207765-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

This is a U.S. Patent Application for:

Title:	MOTION ESTIMATION FOR COMPRESSING MULTIPLE VIEW IMAGES
Inventor:	Niranjan Damera-Venkata
Address:	707 Continental Circle Apt # 1724, Mountain View, CA 94040
Citizenship:	India

EXPRESS MAIL CERTIFICATE OF MAILING

EXPRESS MAIL NO.: ER212308355US

DATE OF DEPOSIT: January 30, 2004

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner for Patents, PO Box 1450, Alexandria, VA 22313-1450.

Edouard Garcia
(Typed or printed name of person mailing paper or fee)

[illegible]
(Signature of person mailing paper or fee)

January 30, 2004
(Date signed)

MOTION ESTIMATION FOR COMPRESSING MULTIPLE VIEW IMAGES

BACKGROUND

In many application environments, an image sequence is encoded (or compressed) to reduce the total amount of data needed to represent the image sequence. The compressed data may then be stored or transmitted more efficiently than the original uncompressed image sequence data. The image sequence may be any sequence of images, including a sequence of video image frames and a sequence of still images. Multiple view image sequences are sequences of images corresponding to different views of a scene; the images may be captured by a single camera positioned at different viewpoints, or the images may be captured by multiple cameras positioned at different locations relative to the scene to capture the scene from different viewpoints.

Image compression methods typically fall into one or more of three main image compression classes: spectral redundancy reduction, spatial redundancy reduction, and temporal redundancy reduction. Spectral redundancy reduction methods typically reduce the amount of image data by discarding spectral data that are not strongly perceived by human eyes. Spatial redundancy reduction methods reduce higher spatial frequency components in the original image data. For example, transform coding is a common spatial redundancy compression method that involves representing an image by a set of transform coefficients. The transform coefficients are quantized individually to reduce the amount of data that is needed to represent the image. A representation of the original image is generated by applying an inverse transform to the transform coefficients. Temporal redundancy reduction methods compress a sequence of images by taking advantage of similarities between successive images. Temporal redundancy may be reduced, for example, by transmitting only those movements or changes in a given image that permit accurate reconstruction of the given image from another image (e.g., a previously received video image frame).

Various different standards of image sequence compression have been developed, often based on block-matching methods. Block-matching methods initially divide a target image (or frame in the case of video image data) to be compressed into an array of blocks (or tiles). Motion data and motion compensation difference data are generated for each block based on a set of data in a reference image (e.g., in a prior video frame) that is similar to the block. In a typical approach, the target image is completely divided into contiguous blocks and sets of pixels in the reference image that best match each block are identified. The target image is reconstructed by accessing and manipulating portions of the reference image. The motion data represents an amount of movement that repositions a suitable part of the reference image to reconstruct a given block of the target image, and the motion-compensated difference data represents intensity adjustments that are made to individual pixels within the set of data from the reference image to accurately reproduce the given block of the target image.

Various methods for computing motion vectors between blocks of a target image and corresponding blocks of a reference image have been proposed. In a typical block matching approach, a current block is compared with all the blocks of like size in a search window superimposed on the reference image. Typically, image blocks of the target image and the reference image are compared by calculating an error function value for each possible match. The motion vector with the smallest error function value is selected as the best matching motion vector for a given target image block. Exemplary block matching error functions are the sum of the absolute values of the differences of the pixels between matched blocks and the sum of the squares of the differences. Motion estimation typically requires a significant portion of the computational resources needed to implement any given image sequence compression method.

SUMMARY

The invention features systems and methods of estimating motion for compressing multiple view images.

In one aspect, the invention features a machine-implemented method of encoding a target image of a scene captured at a first image plane. In accordance with this inventive method, a transformation is computed. The transformation maps at least three noncollinear points substantially coplanar on a scene plane in the target image to corresponding points in a references image of the scene captured at a second image plane different from the first image plane. At least one point in the target image off the scene plane and at least one corresponding point in the reference image are identified. A motion between the target image and the reference image is estimated based on the computed transformation and the identified corresponding off-scene-plane points. The target image is encoded based at least in part on the estimated motion.

The invention also features an apparatus and a machine-readable medium implementing the above-described method.

Other features and advantages of the invention will become apparent from the following description, including the drawings and the claims.

DESCRIPTION OF DRAWINGS

FIG. 1A is a diagrammatic view of a scene of a car being imaged by two cameras positioned at different viewpoints.

FIG. 1B is a diagrammatic view of a scene of a car on a turntable configured to rotate so that a single camera may image the scene from different viewpoints.

FIG. 2 is a block diagram of an encoder encoding a sequence of images and transmitting the encoded image data over a transmission channel, and a decoder decoding the transmitted images to generate a sequence of decoded images.

FIG. 3 is a diagrammatic view of an implementation of a motion estimation engine configured to identify a block of points in a reference image matching a corresponding block in a target image and to compute a motion vector and motion compensation difference data based on the target image block and the identified reference image block.

FIG. 4 is a block diagram of an implementation of a decoder configured to compute a decoded block of a target image based on a reference image, a motion vector computed for the target image block, and motion compensation difference data computed for the target image block.

FIG. 5 is a diagrammatic view of epipolar geometric relationship between a point in a scene imaged by two cameras from different viewpoints.

FIG. 6 is a diagrammatic view of a mapping between an image of a point on a plane in a scene at a first image plane and a corresponding image of the scene plane point at a second image plane.

FIG. 7 is a diagrammatic view of a mapping between an image of an off-scene-plane point at a first image plane and a corresponding image of the off-scene-plane at a second image plane.

FIG. 8 is a flow diagram of an implementation of a method of encoding a target image of a scene.

FIG. 9 is a flow diagram of an implementation of a method of computing a motion vector and motion compensation difference data for block of a target image.

FIG. 10 is a flow diagram of an implementation of a method of computing motion vectors between blocks of a target image and corresponding blocks of a reference image.

DETAILED DESCRIPTION

In the following description, like reference numbers are used to identify like elements. Furthermore, the drawings are intended to illustrate major features of exemplary embodiments in a diagrammatic manner. The drawings are not intended to depict every feature of actual embodiments nor relative dimensions of the depicted elements, and are not drawn to scale.

I. OVERVIEW

The image encoding embodiments described in detail below provide an efficient way to estimate motion between a target image and a reference image that may be incorporated readily into any image compression apparatus or process. These embodiments leverage geometric information computed from certain readily obtainable points in a scene to reduce the size of the search space that is searched to identify points in the reference image corresponding to points in the target image.

FIGS. 1A and 1B show different ways in which multiple view images of a scene may be captured by one or more cameras. As used herein, the term "multiple view images" refers to images of a scene captured from multiple viewpoints. The terms cameras, imaging devices, and image sensors are used interchangeably herein. Exemplary cameras include computer-controllable digital cameras (e.g., a Kodak DCS760 camera), USB video cameras, and Firewire/1394 cameras.

In the implementation of FIG. 1A, images of a scene containing a car 10 are captured by first and second cameras 12, 14 (indicated by their centers C_1, C_2) that are positioned at different viewpoints. The first camera 12 captures images at a scene plane 16 and the second camera 14 captures images at a scene plane 18. As shown in FIG. 1A, point **X** in the scene is imaged at the image planes 16, 18 by a pair of corresponding points $\mathbf{x}_1$, $\mathbf{x}_2$. The points, lines, and planes described herein are represented using homogeneous coordinates. In this representation, the triple of real numbers (x:y:t), with $t \neq 0$, is a set of homogeneous coordinates for a point with Cartesian coordinates (x/t, y/t).

In the implementation of FIG. 1B, images of a scene containing car 10 are captured by a single camera 20 (indicated by its center C_3). The camera 20 may be positioned at different viewpoints relative to the scene. In the illustrated implementation, the position of the camera 20 is fixed and the position of the car 10, which is supported on a turntable 21 that is configured to rotate about an axis 22, changes over time. In this way, the camera 20 is able to capture a sequence of images at an image plane 24 from respective viewpoints that change over time. In other implementations, the position of the car 10 may be fixed, and the position of the camera 20 relative to the car 10 may be changed over time.

FIG. 2 shows multiple view images 30, an encoder 32, a transmission channel 34, a decoder 36, and a set 38 of decoded images. The multiple view images 30 may be video or still images and may be in an analog or a digital format. The pixels of the multiple view images 30 may be presented in an original color model representation (e.g., RGB). The encoder 32 may process the image data in the original color model representation, or encoder 32 may convert the image data to a different color model representation (e.g., YUV) before processing the image data. The encoder 32 includes a motion estimation engine 40, which estimates motion between pairs of multiple view images. Encoder 32 uses the motion estimates to compress the multiple view images 30 by reducing redundancy between the pairs of multiple view images 30. The compressed image data, which includes motion vectors and motion compensation difference data, is transmitted over the transmission channel 34. The transmission channel may be a wired or wireless data transmission connection. The decoder 36 generates decoded images 38 from the image data received over the transmission channel 34 to reconstruct the multiple view images.

FIG. 3 shows an embodiment of motion estimation engine 40 that is configured to compute motion vector and motion compensation difference data 42 for a pair of multiple view images 44, 46. The image 44 to be encoded is referred to herein as the "target image" and the image 46 that is used to encode the target image 44 is referred to herein as the "reference image". These terms are used herein without regard to the temporal ordering (e.g., by capture times or by transmission times) of the images. In this embodiment, the motion estimation engine 40 divides the target image 44 into an array of contiguous, non-overlapping blocks of points. As explained in detail below, for each block 48 motion estimation engine 40 identifies in the reference image 46 a block 50 of points substantially corresponding to the points of block 48. As used herein, a "point" may refer to an individual pixel of an image or may refer to a group of pixels of an image. In this embodiment, there is a one-to-one correspondence between the points of target image block 48 and the points of reference image block 50. As shown in FIG. 3, the reference image block 50 may be deformed (e.g., rotated, translated, or skewed) relative to the target image block 48. Based on the target image block 48 and the corresponding reference image block 50, motion estimation engine 40 computes a motion vector describing movement of the point locations of the reference image block 50 to the locations of the corresponding points of the target image block 48. The motion estimation engine 40 also computes motion compensation difference data that represents intensity adjustments to the points of the reference image block 50 to reproduce the intensity values of the corresponding points of the target image block 48.

FIG. 4 shows an embodiment of decoder 36 that decodes the motion vector and motion compensation difference data 42 computed for a given block of target image 44 based on the reference image 46. In particular, decoder 36 identifies the block in the reference image corresponding to the given target image block using the motion vector. Decoder 36 then adjusts the point values of the identified reference image block using the motion compensation difference data to decode (and thereby reconstruct) the give target image block. This process is repeated for each block of the target image until the entire target image is reconstructed by the decoder 36.

The encoder 32 and the decoder 36 may be incorporated in the same device or in difference devices. In general, encoder 32 and decoder 36 are not limited to any particular hardware or software configuration, but rather they may be implemented in any computing or processing environment, including in digital electronic circuitry or in computer hardware, firmware, device driver, a codec (e.g., an MPEG video codec), or software. The encoder 32 and the decoder 36 may be embedded in the hardware of any one of a wide variety of electronic devices, including desktop computers, laptop computers, and portable electronic devices (e.g., digital still image camera, digital video cameras, mobile phones and personal digital assistants). In some implementations, each of encoder 32 and decoder 36 is implemented, at least in part, in a computer process product tangibly embodied in a machine-readable storage device for execution by a computer processor. In some embodiments, encoder 32 and decoder 36 preferably are implemented in a high level procedural or object oriented processing language; however, the algorithms may be implemented in assembly or machine language, if desired. In any case, the processing language may be a compiled or interpreted language. Suitable processors include, for example, both general and special purpose microprocessors. Generally, a processor receives instructions and data from a read-only memory and/or a random access memory. Storage devices suitable for tangibly embodying computer process instructions include all forms of non-volatile memory, including, for example, semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM. Any of the foregoing technologies may be supplemented by or incorporated in specially designed ASICs (application-specific integrated circuits).

II. ESTIMATING MOTION BETWEEN PAIRS OF MULTIPLE VIEW IMAGES

As mentioned above, the image encoding embodiments described herein provide an efficient way to estimate motion between a target image and a reference image that leverages geometric information computed from certain readily obtainable points in a scene to reduce the size of the search space that is searched to identify points in the reference image corresponding to points in the target image. FIGS. 5, 6, and 7 show the geometric relationships between a pair of multiple view images that are leveraged by the image encoding embodiments described herein.

FIG. 5 shows the point correspondence geometry for a pair of cameras 12, 14 with respective centers C_1, C_2 and respective image planes 16, 18. As shown, the camera centers C_1, C_2, the 3-space point $\mathbf{X}$, and the images $\mathbf{x}_1$, $\mathbf{x}_2$ of the point $\mathbf{X}$ in image planes 16, 18 lie in a common plane 54, which is referred to herein as an "epipolar plane". Upper case variables are used herein to represent three-dimensional world coordinates, and lower case variables are used to represent two-dimensional image plane coordinates. A camera baseline 56 connects the centers C_1, C_2 of cameras 12, 14, and intersects the image planes 16, 18 at respective points $\mathbf{e}_1$, $\mathbf{e}_2$, which are referred to herein as "epipoles".

Referring to FIG. 6, a ray 58 that extends through the first camera center C_1 and point $\mathbf{x}_1$ in image plane 16 meets a plane 60 (labeled π), which does not pass through either of the two camera centers C_1, C_2, at a point $\mathbf{X}_\pi$. Point $\mathbf{X}_\pi$ projects to a point $\mathbf{x}_2$ in the second image plane 18. Since $\mathbf{X}_\pi$ lies on the ray 58 corresponding to $\mathbf{x}_1$, the projected point $\mathbf{x}_2$ must lie on the epipolar line $\mathbf{l}_2$ corresponding to the image of ray 58 in image plane 18. The points $\mathbf{x}_1$, $\mathbf{x}_2$ are images of the three-dimension point $\mathbf{X}_\pi$ lying on plane 60. The set of all such points $\mathbf{x}^{(i)}_1$ in the first image plane 16 and the corresponding points $\mathbf{x}^{(i)}_2$ in the second image plane 18 are related by a two-dimensional homography H_π that maps each $\mathbf{x}^{(i)}_1$ to $\mathbf{x}^{(i)}_2$. That is,

$$\mathbf{x}^{(i)}_1 = H_\pi \cdot \mathbf{x}^{(i)}_2 \tag{1}$$

where

$$H_\pi = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \tag{2}$$

Since H_π has eight degrees of freedom up to scale, H_π may be determined from four sets of corresponding points, $\mathbf{x}^{(i)}_1 \leftrightarrow \mathbf{x}^{(i)}_2$, where i has an integer value from 1 to 4. In some implementations, more than four sets of correspondence points may be used in a singular value decomposition process to determine a least-squares solution for H.

FIG. 7 shows a point $\mathbf{X}$ that is not on the plane 60. A ray through $\mathbf{X}$ intersects the plane 60 at the point $\mathbf{X}_\pi$. The images of $\mathbf{X}$ and $\mathbf{X}_\pi$ are coincident points at $\mathbf{x}_1$ in the first image plane 16. In the second image plane, $\mathbf{X}$ and $\mathbf{X}_\pi$ are imaged at $\mathbf{x}_2$ and $\mathbf{x'}_2 = \mathrm{H}_\pi \mathbf{x}_1$, respectively. The points $\mathbf{x}_2$ and $\mathbf{x'}_2$ both lie on the epipolar line $\mathbf{l}_2$. The vector between the points $\mathbf{x}_2$ and $\mathbf{x'}_2$ is the parallax relative to the homography induced by the plane 60. In this geometry, the epipolar line $\mathbf{l}^{(x)}_2$ is given by:

$$\mathbf{l}^{(x)}_2 = \mathbf{x'}_2 \times \mathbf{x}_2 \tag{3}$$

where $\times$ is the cross product operator. Since $\mathbf{x'}_2 = \mathrm{H}_\pi \mathbf{x}_1$,

$$\mathbf{l}^{(x)}_2 = (\mathrm{H}_\pi \mathbf{x}_1) \times \mathbf{x}_2 \tag{4}$$

Similarly, the epipolar line $\mathbf{l}^{(y)}_2$ is given by:

$$\mathbf{l}^{(y)}_2 = \mathbf{y'}_2 \times \mathbf{y}_2 \tag{5}$$

and since $\mathbf{y'}_2 = \mathrm{H}_\pi \mathbf{y}_1$,

$$\mathbf{l}^{(y)}_2 = (\mathrm{H}_\pi \mathbf{y}_1) \times \mathbf{y}_2 \tag{6}$$

The epipole $\mathbf{e}_2$ may be computed from $\mathbf{l}^{(x)}_2 \times \mathbf{l}^{(y)}_2$, **which are given by** equations (4) and (6), if the corresponding off-scene-plane points $\mathbf{x}_1 \leftrightarrow \mathbf{x}_2$ and $\mathbf{y}_1 \leftrightarrow \mathbf{y}_2$ are known. The epipole $\mathbf{e}_2$ then may be used to determine epipolar lines in the image plane 18. With this information, any given point $\mathbf{w}_2$ in the second image plane 18 corresponds to the homographic projection of a corresponding point $\mathbf{w}_1$ in the first image plane 16 plus a possible displacement along the epipolar line $\mathbf{l}^{(w)}_2$. That is,

$$\mathbf{w}_2 = (\mathrm{H}_\pi \mathbf{w}_1) + \Theta \cdot \mathbf{e}_2 \tag{7}$$

where Θ is a scalar.

In some implementations, the full projection model of equation (2) is approximated by an affine approximation model. The accuracy of this model improves as the distances of points in the scene to the centers of the camera(s) become much greater than the distances between points in the scene. In this model, the transformation is given by:

$$\mathrm{H}_\pi = \begin{bmatrix} \mathrm{H}_{11} & \mathrm{H}_{12} & \mathrm{H}_{13} \\ \mathrm{H}_{21} & \mathrm{H}_{22} & \mathrm{H}_{23} \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

Since H_π has six degrees of freedom up to scale in the affine approximation model, H_π may be determined from three sets of corresponding points, $\mathbf{x}^{(i)}_1 \leftrightarrow$

$\mathbf{x}^{(i)}_2$, where i has an integer value from 1 to 3. In the affine approximation model, the epipolar lines are assumed to be parallel, in which case only a single epipolar line is needed to determine the epipole $\mathbf{e}_2$. Indeed, if an epipolar line $\mathbf{l}_2$ = (a,b,c), then the corresponding epipole $\mathbf{e}_2$ = (-b,a,0). The epipolar line $\mathbf{l}_2$ may be determined from images of a single off-scene-plane point (e.g., point **X**).

FIG. 8 shows an embodiment of a method by which encoder 32 encodes multiple view images of a scene based on the geometric relationships described above in connection with FIGS. 5-7. In this method, the encoder 32 computes a transformation (e.g., H_π) (step 68). Encoder 32 computes the transformation based on three pairs of corresponding scene-plane points by solving for the coefficients of the transformation matrix in equation (2).

Referring back to FIGS. 1A and 1B, the transformation H_π is computed from images of at least three noncollinear points 70, 72, 74 (FIG. 1A) or 76, 78, 80 (FIG. 1B) that are substantially coplanar on a planar surface 82 (FIG. 1A) or 84 (FIG. 1B) in the scene. If the full projection model of equation (2) is used, at least four nonlinear, substantially coplanar points in the scene are used to compute H_π. If the above-described affine approximation model is used, at least three nonlinear, substantially coplanar points in the scene are used to compute H_π. The transformation H_π maps the noncollinear, substantially coplanar points imaged in the target image to corresponding points imaged in the reference image of the scene. The scene planes 82, 84 may correspond to any planar surface in the scene. In the implementation illustrated in FIG. 1A, the scene plane 82 corresponds to a known checkerboard pattern and the noncollinear, coplanar scene plane points 70-74 correspond three corner points of one of the checkerboard squares. In the implementation illustrated in FIG. 1B, the scene plane 84 corresponds to the top surface of turntable 21, which includes a regular pattern. The checkerboard pattern on scene plane 82 and the regular pattern on scene plane 84 enable the pairs of corresponding points in image planes 16, 18 to be easily and accurately determined. In other implementations, point correspondences may be determined in another way (e.g., by projecting known light patterns onto a planar surface in the scene). The correspondences between scene-plane points may be determined automatically using any automatic point correspondence matching process. Alternatively, a user may identify the correspondences between scene-plane points manually.

Encoder 32 also identifies at least one point (e.g., point **X**) in the scene that does not lie on the planar surface used to compute the transformation H_π. This point is imaged by a point (e.g., point $\mathbf{x}_1$) in the target image plane 16 and a corresponding point (e.g., point $\mathbf{x}_2$) in the reference image plane 18 (step 86). Only pne pair of corresponding off-scene-plane points is used in the affine approximation model, whereas two such pairs of points are used in the full projection model of equation (2). The off-scene-plane points may be determined automatically using any automatic point correspondence matching process, or a user may identify the corresponding off-scene-plane points manually.

Based on the computed transformation and the identified corresponding off-scene-plane points, the encoder 32 estimates motion between the target image and the reference image (step 88). In some implementations, using equations (4) and (6) the encoder 32 computes the epipole $\mathbf{e}_2$ in the reference image plane 18 based on the identified corresponding off-scene-plane points. The epipole $\mathbf{e}_2$ is used to reduce the search space for blocks in the reference image that match corresponding blocks in the target image to a single dimension, as explained in detail below.

Referring to FIGS. 9 and 10, in some embodiments, encoder 32 computes respective motion vectors for blocks of points in the target image. In this regard, encoder 32 divides the target image into an array of contiguous, non-overlapping blocks of points. (step 90). In other implementations, encoder 32 may compute motion vectors for nodes in a mesh-based representation of the target image.

For each block, the encoder 32 computes a respective motion vector representing motion between a target image block and a reference image block (step 92). In this process, the encoder selects the next block of the target image as the next block (step 94). For each target image block, the encoder defines a single-parameter search space relating the target image points to a block of points in the reference image (step 96). The single-parameter search space is defined with respect to the following motion model, which is derived from equation (7):

$$\mathbf{b}_i' = (H_\pi \cdot \mathbf{b}_i) + \Theta \cdot \mathbf{e}_2 \qquad (9)$$

where H_π is the computed transformation, Θ is a scalar, $\mathbf{e}_2$ is the respective epipole, $\mathbf{b}_i$ is any point within the target image block being encoded, and $\mathbf{b}_i'$ is the corresponding point in the reference image lying on the epipolar line $\mathbf{l}_i$. In this way, the search space for the points within block $\mathbf{b}_i'$ in the reference image is parameterized by a single parameter Θ. The encoder 32 searches each single-parameter search space for a block of points in the reference image corresponding to the current target image block (step 98). This process is repeated for each block in the target image (step 100)

In some implementations, a matching score is generated for each of multiple different values for Θ and the block $\mathbf{b}_i'$ corresponding to the value of Θ producing the highest matching score is selected as the reference image block corresponding to the target image block. The matching score may be generated based on a correlation matching measure (e.g., the sum of the absolute values of the differences of the pixels between matched blocks, or the sum of the squares of the differences) or to some other point match (or correlation) measure. The value of Θ defines a motion vector since specification of Θ determines the reference image block matching the corresponding target image block.

For each block of the target image and for each computed motion vector, the encoder 32 also computes motion compensation difference data (step 102). The motion compensation difference data represents intensity adjustments to the points of the reference image block needed to reproduce the intensity values of the points of the target image block.

Referring back to FIG. 8, after the motion between the target image and the reference image has been estimated (step 88), the encoder 32 encodes the target image based at least in part on the estimated motion (step 104). In some implementations, encoder 32 may encode the target image in accordance with any known predictive coding technique. For example, in some implementations, encoder 32 may encode the target image in accordance with a motion compensation prediction approach. Encoder 32 also may incorporate non-motion-vector-based image compression methods in the process of encoding the target image, including spectral and spatial redundancy reduction methods. In addition, encoder 32 may use any color interleaving process to encode the target image before transmitting the encoded data over transmission channel 34.

Other embodiments are within the scope of the claims.